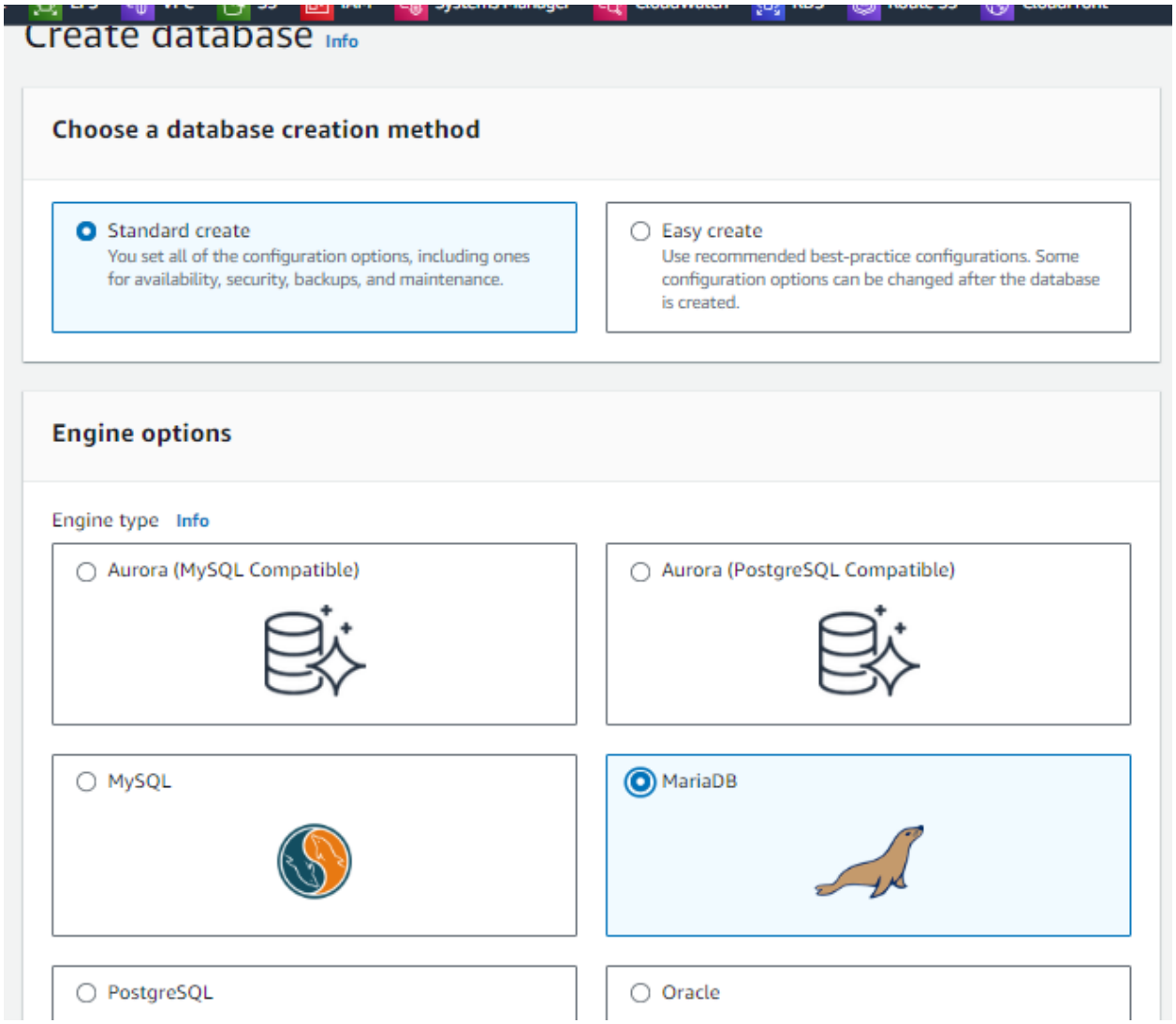
First of all we have to create an RDS database



DB instance identifier Info

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ **Credentials Settings**

Master username Info

Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. The first character must be a letter.

Credentials management

You can use AWS Secrets Manager or manage your master user credentials.

○ Managed in AWS Secrets Manager - *most secure*
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

◉ Self managed
Create your own password or have RDS create a password that you manage.

☐ Auto generate password
Amazon RDS can generate a password for you, or you can specify your own password.

Master password Info

••••••••••

Password strength Strong

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' " @

Confirm master password Info

Storage type Info

Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp3)
Performance scales independently from storage

Allocated storage Info

25 GiB

Minimum: 20 GiB. Maximum: 65,536 GiB

After you modify the storage for a DB instance, the status of the DB instance will be in storage-optimization. Your instance will remain available as the storage-optimization operation completes. Learn more

Advanced settings

Baseline IOPS of 3,000 IOPS and storage throughput of 125 MiBps are included for allocated storage less than 400 GiB.

Storage autoscaling

Availability & durability

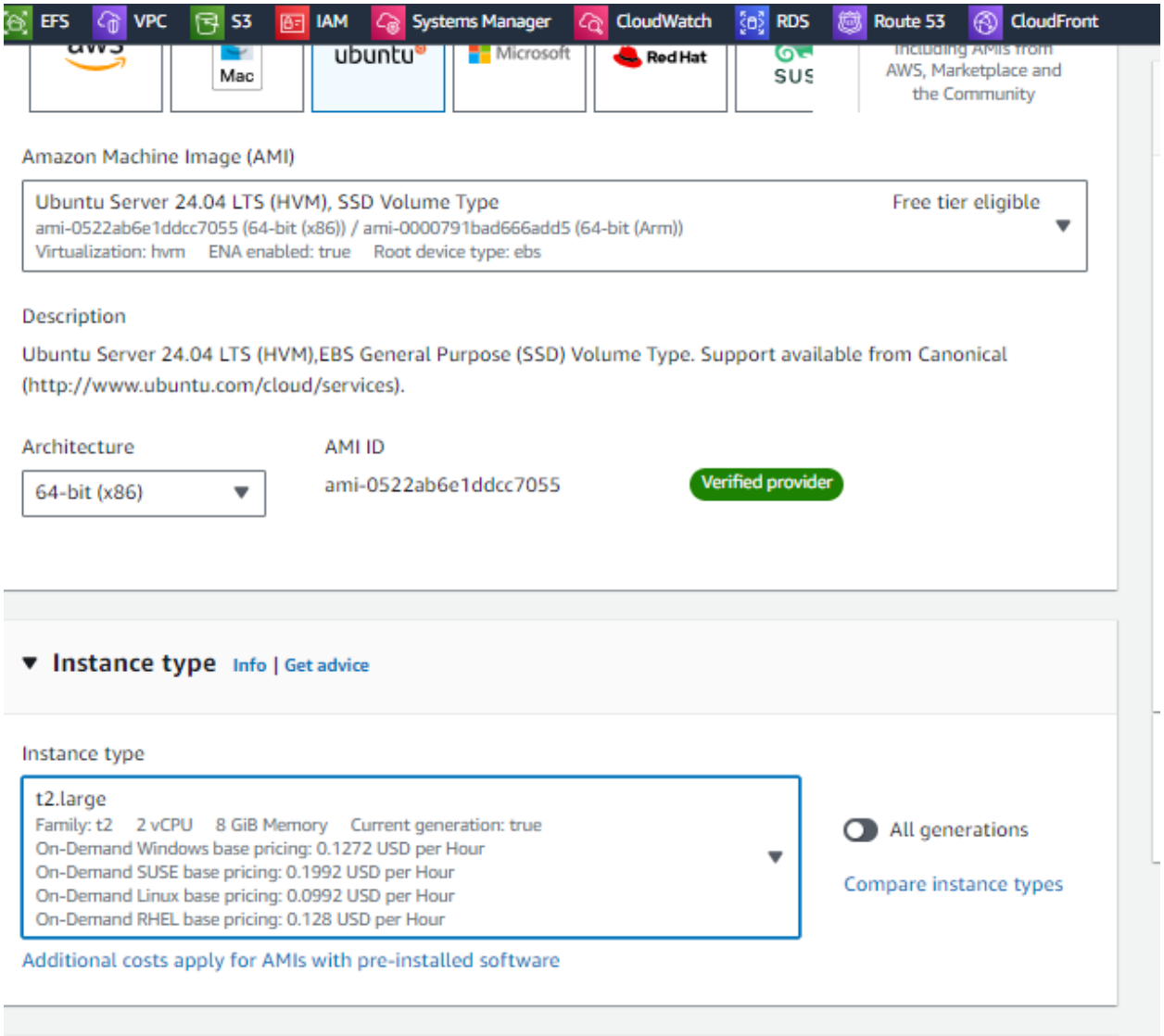
Multi-AZ deployment Info

Create a standby instance (recommended for production usage)
Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

Do not create a standby instance

Feedback

Create an instance with t2 large



After creating database git clone the repository for project

Git clone https://github.com/rajatpzade/anguler-java.git

```
ubuntu@ip-172-31-10-55:~$ sudo -i
root@ip-172-31-10-55:~# git clone https://github.com/rajatpzade/anguler-java.git
Cloning into 'anguler-java'...
remote: Enumerating objects: 80, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 80 (delta 3), reused 80 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (80/80), 268.11 KiB | 14.89 MiB/s, done.
Resolving deltas: 100% (3/3), done.
root@ip-172-31-10-55:~#
```

Then run the command for installation of mariadb-server

```
sudo apt update
sudo apt install mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

```
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-10-55:~/anguler-java# systemctl start mariadb
root@ip-172-31-10-55:~/anguler-java# systemctl enable mariadb
Synchronizing state of mariadb.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mariadb
root@ip-172-31-10-55:~/anguler-java#
```

CloudShell Feedback

CREATE DATABASE springbackend;

GRANT ALL PRIVILEGES ON springbackend.* TO 'admin'@'13.201.190.80' IDENTIFIED BY 'Pratham123';

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE springbackend;
Query OK, 1 row affected (0.005 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON springbackend.* TO 'admin'@'13.201.190.80' IDENTIFIED BY 'Pratham123';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]>
```

Run the databases query provided by developer.

```
MariaDB [springbackend]> COMMIT;
Query OK, 0 rows affected (0.001 sec)

MariaDB [springbackend]> show tables
    -> ;
+-------------------------+
| Tables_in_springbackend |
+-------------------------+
| tbl_workers             |
+-------------------------+
1 row in set (0.001 sec)

MariaDB [springbackend]> select * from tbl_workers
    -> ;
+----+----------+-------------+-------------+
| id | status   | workerfname | workerlname |
+----+----------+-------------+-------------+
|  1 | Working  | Ivan        | Holicek     |
| 37 | Vacation | Marko       | Markovic    |
| 40 | Working  | Ivo         | Ivica       |
| 41 | Working  | Luka        | Lukovic     |
| 42 | Working  | Filip       | Filipovic   |
+----+----------+-------------+-------------+
5 rows in set (0.004 sec)

MariaDB [springbackend]>
```

Now you have to give your RDS endpoint, user and password in application.properties file/

```
EC2  EFS  VPC  S3  IAM  Systems Manager  CloudWatch  RDS  Route 53  CloudFront
spring.datasource.url=jdbc:mysql://database-1.ctq2s4eqkzog.ap-south-1.rds.amazonaws.com:3306/springbackend?useSSL=false
spring.datasource.username=admin
spring.datasource.password=Pratham123

spring.jpa.generate-ddl=true
~
~
~
~
```

```
root@ip-172-31-10-55:~/anguler-java# ls
README.md  angular-frontend  spring-backend  springbackend.sql
root@ip-172-31-10-55:~/anguler-java# cd spring-backend/
root@ip-172-31-10-55:~/anguler-java/spring-backend# cd
root@ip-172-31-10-55:~# vim docker.sh
root@ip-172-31-10-55:~# chmod o+x docker.sh
root@ip-172-31-10-55:~# ./docker.sh
```

Then install a docker using the script & run it

Vim docker.sh

Chmod o+x docker.sh

./docker.sh

Then create a dockerfile for backend-server as below

```
# Use Ubuntu as base image
FROM ubuntu:latest

# Install dependencies
RUN apt-get update && \
    apt-get install -y openjdk-8-jdk maven && \
    rm -rf /var/lib/apt/lists/*

# Set the working directory in the container
WORKDIR /app

# Copy the Maven project directory into the container
COPY . /app

# Build the Maven project
RUN mvn clean package -Dmaven.test.skip=true

# Expose the port the application runs on
EXPOSE 8080

# Command to run the application
CMD ["java", "-jar", "target/spring-backend-v1.jar"]
~
~
~
~
~
```

After that build the image using

Docker build . –t spring:backend

```
README.md  dockerfile  mvnw  mvnw.cmd  pom.xml  src
root@ip-172-31-10-55:~/anguler-java/spring-backend# docker build . -t spring:backend
[+] Building 91.0s (10/10) FINISHED
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 553B
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/ubuntu:latest@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee
 => => resolve docker.io/library/ubuntu:latest@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee
 => => sha256:31e907dcc94a592a57796786399eb004dcbba714389fa615f5efa05a91316356 29.71MB / 29.71MB
 => => sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee 1.34kB / 1.34kB
 => => sha256:d35dfc2fe3ef66bcc085ca00d3152b482e6cafb23cdda1864154caf3b19094ba 424B / 424B
 => => sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a 2.30kB / 2.30kB
 => => extracting sha256:31e907dcc94a592a57796786399eb004dcbba714389fa615f5efa05a91316356
 => [internal] load build context
 => => transferring context: 87.03kB
 => [2/5] RUN apt-get update &&     apt-get install -y openjdk-8-jdk maven &&     rm -rf /var/lib/apt/lists/*
 => [3/5] WORKDIR /app
 => [4/5] COPY . /app
 => [5/5] RUN mvn clean package -Dmaven.test.skip=true
 => exporting to image
 => => exporting layers
 => => writing image sha256:f5d1774561291988f41ae0e863f5b2ac63e7ef7df591ad8459935d7004fd02fb
 => => naming to docker.io/library/spring:backend
root@ip-172-31-10-55:~/anguler-java/spring-backend# docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
spring       backend   f5d177456129   About a minute ago   823MB
root@ip-172-31-10-55:~/anguler-java/spring-backend#
```

After that check the docker image

Then create a container using docker image & assign port number

Docker run –d –p 8080:8080 spring :backend

```
 => => naming to docker.io/library/spring:backend
root@ip-172-31-10-55:~/anguler-java/spring-backend# docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
spring       backend   f5d177456129   About a minute ago   823MB
root@ip-172-31-10-55:~/anguler-java/spring-backend# docker run -d -p 8080:8080 spring:backend
c07bc4042b2836352fc4ddaf050b19311519fe7a23e8f088759e7fe8e89e39a2
root@ip-172-31-10-55:~/anguler-java/spring-backend# docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS                                       NAMES
c07bc4042b28   spring:backend   "java -jar target/sp…"   4 seconds ago   Up 3 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   sharp_swanson
root@ip-172-31-10-55:~/anguler-java/spring-backend#
```

Now go to frontend & here we have to mention our backend ip in worker.service.ts file

Here also we have to create a docker file for Frontend server as below.

```
# Use official Node.js image as the base image
FROM node:14-alpine as build

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json (if available)
COPY package*.json ./

# Install project dependencies
RUN npm install

# Copy the rest of the application code
COPY . .

# Build the Angular application
RUN npm run build

# Use NGINX as the production server
FROM nginx:alpine

# Copy the built artifact from the previous stage to NGINX web server directory
COPY --from=build /usr/src/app/dist/angular-frontend /usr/share/nginx/html

# Expose port 80 to the outside world
EXPOSE 80

# Start NGINX server when the container starts
CMD ["nginx", "-g", "daemon off;"]
root@ip-172-31-10-55:~/anguler-java/angular-frontend#
```

After that build the image using

Docker build . –t angular:frontend

```
root@ip-172-31-10-55:~/anguler-java/angular-frontend# cd src/
root@ip-172-31-10-55:~/anguler-java/angular-frontend/src# ls
app  assets  environments  favicon.ico  index.html  main.ts  polyfills.ts  styles.css  test.ts
root@ip-172-31-10-55:~/anguler-java/angular-frontend/src# cd app/
root@ip-172-31-10-55:~/anguler-java/angular-frontend/src/app# ls
app.component.css  app.component.html  app.component.spec.ts  app.component.ts  app.module.ts  components  models  services
root@ip-172-31-10-55:~/anguler-java/angular-frontend/src/app# cd services/
root@ip-172-31-10-55:~/anguler-java/angular-frontend/src/app/services# ls
worker.service.ts
root@ip-172-31-10-55:~/anguler-java/angular-frontend/src/app/services# vim worker.service.ts
root@ip-172-31-10-55:~/anguler-java/angular-frontend/src/app/services# cd
root@ip-172-31-10-55:~# cd anguler-java/
root@ip-172-31-10-55:~/anguler-java# ls
README.md  angular-frontend  spring-backend  springbackend.sql
root@ip-172-31-10-55:~/anguler-java# cd angular-frontend/
root@ip-172-31-10-55:~/anguler-java/angular-frontend# ls
README.md  angular.json  dockerfile  karma.conf.js  package-lock.json  package.json  src  tsconfig.app.json  tsconfig.json  tsconfig.spec.json
root@ip-172-31-10-55:~/anguler-java/angular-frontend# vim dockerfile
root@ip-172-31-10-55:~/anguler-java/angular-frontend# docker build . -t angular:frontend
[+] Building 45.4s (11/13)
 => WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 2)
 => [internal] load build context
 => => transferring context: 820.19kB
 => [stage-1 1/2] FROM docker.io/library/nginx:alpine@sha256:a5127daff3d6f4606be3100a252419bfa84fd6ee5cd74d0feaca1a5068f97dcf
```

After that check the docker image

Then create a container using docker image & assign port number

Docker run –d –p 80:80 angular:frontend

```
1 warning found (use docker --debug to expand):
 - FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 2)
root@ip-172-31-10-55:~/anguler-java/angular-frontend# docker images
REPOSITORY   TAG        IMAGE ID       CREATED              SIZE
angular      frontend   3ce1a8714b4c   About a minute ago   43.5MB
spring       backend    f5d177456129   11 minutes ago       823MB
root@ip-172-31-10-55:~/anguler-java/angular-frontend# docker run -d -p 80:80 angular:frontend
b89342701a4eb23161a37cce675e50472121fa69992d45a589bb2e11a5cf63f6
root@ip-172-31-10-55:~/anguler-java/angular-frontend# docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS         PORTS                                       NAMES
b89342701a4e   angular:frontend   "/docker-entrypoint.…"   4 seconds ago    Up 3 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp           quirky_brahmagupta
c07bc4042b28   spring:backend     "java -jar target/sp…"   10 minutes ago   Up 9 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   sharp_swanson
root@ip-172-31-10-55:~/anguler-java/angular-frontend#
```

Output :- http://13.201.190.80:80